ALGORITHMIQUE ExerciceS

Sommaire

PARTIE 1 Enonce des Exercices

Exercice 1.1

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B en Entier
Début
A ← 1
B ← A + 3
A ← 3
Fin
```

Exercice 1.2

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C en Entier
Début
A ← 5
B ← 3
C ← A + B
A ← 2
C ← B – A
Fin
```

Exercice 1.3

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B en Entier
Début
A ← 5
B ← A + 4
A ← A + 1
B ← A – 4
Fin
```

Exercice 1.4

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C en Entier
Début
 A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C
Fin
```

Exercice 1.5

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B en Entier
Début
A ← 5
B ← 2
A ← B
B ← A
Fin
```

Moralité : les deux dernières instructions permettent-elles d'échanger les deux valeurs de B et A ? Si l'on inverse les deux dernières instructions, cela change-t-il quelque chose ?

Exercice 1.6

Plus difficile, mais c'est un classique absolu, qu'il faut absolument maîtriser : écrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable.

Exercice 1.7

Une variante du précédent : on dispose de trois variables A, B et C. Ecrivez un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (toujours quels que soient les contenus préalables de ces variables).

Exercice 1.8

Que produit l'algorithme suivant ?

```
Variables A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A + B
Fin
```

Exercice 1.9

Que produit l'algorithme suivant ?

```
Variables A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A & B
Fin
```

Corrigés des Exercices

Exercice 1.1

```
Après         La valeur des variables est :
A ← 1         A = 1          B = ?
B ← A + 3     A = 1          B = 4
A ← 3         A = 3          B = 4
```

Exercice 1.2

```
Après         La valeur des variables est :
A ← 5         A = 5          B = ?          C = ?
B ← 3         A = 5          B = 3          C = ?
C ← A + B     A = 5          B = 3          C = 8
A ← 2         A = 2          B = 3          C = 8
C ← B – A     A = 2          B = 3          C = 1
```

Exercice 1.3

```
Après         La valeur des variables est :
A ← 5         A = 5          B = ?
B ← A + 4     A = 5          B = 9
A ← A + 1     A = 6          B = 9
B ← A – 4     A = 6          B = 2
```

Exercice 1.4

```
Après         La valeur des variables est :
A ← 3         A = 3          B = ?          C = ?
B ← 10        A = 3          B = 10         C = ?
C ← A + B     A = 3          B = 10         C = 13
B ← A + B     A = 3          B = 13         C = 13
A ← C         A = 13         B = 13         C = 13
```

Exercice 1.5

```
Après         La valeur des variables est :
A ← 5         A = 5          B = ?
B ← 2         A = 5          B = 2
A ← B         A = 2          B = 2
B ← A         A = 2          B = 2
```

Les deux dernières instructions ne permettent donc pas d'échanger les deux valeurs de B et A, puisque l'une des deux valeurs (celle de A) est ici écrasée.
Si l'on inverse les deux dernières instructions, cela ne changera rien du tout, hormis le fait que cette fois c'est la valeur de B qui sera écrasée.

Exercice 1.6

```
Début
…
C ← A
A ← B
B ← C
Fin
```

On est obligé de passer par une variable dite temporaire (la variable C).

Exercice 1.7

```
Début
…
D ← C
C ← B
```

```
B ← A
A ← D
Fin
```

En fait, quel que soit le nombre de variables, une seule variable temporaire suffit…

Exercice 1.8

Il ne peut produire qu'une erreur d'exécution, puisqu'on ne peut pas additionner des caractères.

Exercice 1.9

…En revanche, on peut les concaténer. A la fin de l'algorithme, C vaudra donc "42312".



PARTIE 2
Enonce des Exercices

Exercice 2.1

Quel résultat produit le programme suivant ?

```
Variables val, double numériques
Début
Val ← 231
Double ← Val * 2
Ecrire Val
Ecrire Double
Fin
```

Exercice 2.2

Ecrire un programme qui demande un nombre à l'utilisateur, puis qui calcule et affiche le carré de ce nombre.

Exercice 2.3

Ecrire un programme qui lit le prix HT d'un article, le nombre d'articles et le taux de TVA, et qui fournit le prix total TTC correspondant. Faire en sorte que des libellés apparaissent clairement.

-

Exercice 2.4

Ecrire un algorithme utilisant des variables de type chaîne de caractères, et affichant quatre variantes possibles de la célèbre « belle marquise, vos beaux yeux me font mourir d'amour ». On ne se soucie pas de la ponctuation, ni des majuscules.

Corrigés des Exercices

Exercice 2.1

On verra apparaître à l'écran 231, puis 462 (qui vaut 231 * 2)

Exercice 2.2

```
Variables nb, carr en Entier
Début
Ecrire "Entrez un nombre :"
Lire nb
carr ← nb * nb
Ecrire "Son carré est : ", carr
Fin
```

En fait, on pourrait tout aussi bien économiser la variable carr en remplaçant les deux avant-dernières lignes par :

```
Ecrire "Son carré est : ", nb*nb
```

C'est une question de style ; dans un cas, on privilégie la lisibilité de l'algorithme, dans l'autre, on privilégie l'économie d'une variable.

Exercice 2.3

```
Variables nb, pht, ttva, pttc en Numérique
Début
Ecrire "Entrez le prix hors taxes :"
Lire pht
Ecrire "Entrez le nombre d'articles :"
Lire nb
Ecrire "Entrez le taux de TVA :"
Lire ttva
pttc ← nb * pht * (1 + ttva)
Ecrire "Le prix toutes taxes est : ", pttc
Fin
```

Là aussi, on pourrait squeezer une variable et une ligne en écrivant directement. :

```
Ecrire "Le prix toutes taxes est : ", nb * pht * (1 + ttva)
```

C'est plus rapide, plus léger en mémoire, mais un peu plus difficile à relire (et à écrire !)

Exercice 2.4

```
Variables t1, t2, t3, t4 en Caractère
Début
t1 ← "belle Marquise"
t2 ← "vos beaux yeux"
t3 ← "me font mourir"
t4 ← "d'amour"
Ecrire t1 & " " & t2 & " " & t3 & " " & t4
Ecrire t3 & " " & t2 & " " & t4 & " " & t1
Ecrire t2 & " " & t3 & " " & t1 & " " & t4
Ecrire t4 & " " & t1 & " " & t2 & " " & t3
Fin
```

PARTIE 3
Enonce des Exercices

Exercice 3.1

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on laisse de côté le cas où le nombre vaut zéro).

Exercice 3.2

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de côté le cas où le produit est nul). Attention toutefois : on ne doit **pas** calculer le produit des deux nombres.

Exercice 3.3

Ecrire un algorithme qui demande trois noms à l'utilisateur et l'informe ensuite s'ils sont rangés ou non dans l'ordre alphabétique.

Exercice 3.4

Ecrire un algorithme qui demande un nombre à l'utilisateur, et l'informe ensuite si ce nombre est positif ou négatif (on inclut cette fois le traitement du cas où le nombre vaut zéro).

Exercice 3.5

Ecrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si le produit est négatif ou positif (on inclut cette fois le traitement du cas où le produit peut être nul). Attention toutefois, on ne doit pas calculer le produit !

Exercice 3.6

Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- "Poussin" de 6 à 7 ans
- "Pupille" de 8 à 9 ans
- "Minime" de 10 à 11 ans
- "Cadet" après 12 ans

Peut-on concevoir plusieurs algorithmes équivalents menant à ce résultat ?

Corrigés des Exercices

Exercice 3.1

```
Variable n en Entier
Début
Ecrire "Entrez un nombre : "
Lire n
Si n > 0 Alors
  Ecrire "Ce nombre est positif”
```

```
Sinon
  Ecrire "Ce nombre est négatif"
Finsi
Fin
```

Exercice 3.2

```
Variables m, n en Entier
Début
Ecrire "Entrez deux nombres : "
Lire m, n
Si (m > 0 ET n > 0) OU (m < 0 ET n < 0) Alors
  Ecrire "Leur produit est positif"
Sinon
  Ecrire "Leur produit est négatif"
Finsi
Fin
```

Exercice 3.3

```
Variables a, b, c en Caractère
Début
Ecrire "Entrez successivement trois noms : "
Lire a, b, c
Si a < b ET b < c Alors
  Ecrire "Ces noms sont classés alphabétiquement"
Sinon
  Ecrire "Ces noms ne sont pas classés"
Finsi
Fin
```

Exercice 3.4

```
Variable n en Entier
Début
Ecrire "Entrez un nombre : "
Lire n
Si n < 0 Alors
  Ecrire "Ce nombre est négatif"
SinonSi n = 0 Alors
  Ecrire "Ce nombre est nul"
Sinon
  Ecrire "Ce nombre est positif"
Finsi
Fin
```

Exercice 3.5

```
Variables m, n en Entier
Début
Ecrire "Entrez deux nombres : "
Lire m, n
Si m = 0 OU n = 0 Alors
  Ecrire "Le produit est nul"
SinonSi (m < 0 ET n < 0) OU (m > 0 ET n > 0) Alors
  Ecrire "Le produit est positif"
Sinon
  Ecrire "Le produit est négatif"
```

```
Finsi
Fin
```

Si on souhaite simplifier l'écriture de la condition lourde du SinonSi, on peut toujours passer par des variables booléennes intermédiaires. Une astuce de sioux consiste également à employer un Xor (c'est l'un des rares cas dans lesquels il est pertinent)

Exercice 3.6

```
Variable age en Entier
Début
Ecrire "Entrez l'âge de l'enfant : "
Lire age
Si age >= 12 Alors
  Ecrire "Catégorie Cadet"
SinonSi age >= 10 Alors
  Ecrire "Catégorie Minime"
SinonSi age >= 8 Alors
  Ecrire "Catégorie Pupille"
SinonSi age >= 6 Alors
  Ecrire "Catégorie Poussin"
Finsi
Fin
```

On peut évidemment écrire cet algorithme de différentes façons, ne serait-ce qu'en commençant par la catégorie la plus jeune.



PARTIE 4
Enonce des Exercices

Exercice 4.1

Formulez un algorithme équivalent à l'algorithme suivant :

```
Si Tutu > Toto + 4 OU Tata = "OK" Alors
  Tutu ← Tutu + 1
Sinon
  Tutu ← Tutu – 1
Finsi
```

Exercice 4.2

Cet algorithme est destiné à prédire l'avenir, et il doit être infaillible !

Il lira au clavier l'heure et les minutes, et il affichera l'heure qu'il sera une minute plus tard. Par exemple, si l'utilisateur tape 21 puis 32, l'algorithme doit répondre :

"Dans une minute, il sera 21 heure(s) 33".

NB : on suppose que l'utilisateur entre une heure valide. Pas besoin donc de la vérifier.

Exercice 4.3

De même que le précédent, cet algorithme doit demander une heure et en afficher une autre. Mais cette fois, il doit gérer également les secondes, et afficher l'heure qu'il sera une seconde plus tard.

Par exemple, si l'utilisateur tape 21, puis 32, puis 8, l'algorithme doit répondre : "Dans une seconde, il sera 21 heure(s), 32 minute(s) et 9 seconde(s)".

NB : là encore, on suppose que l'utilisateur entre une date valide.

Exercice 4.4

Un magasin de reprographie facture 0,10 E les dix premières photocopies, 0,09 E les vingt suivantes et 0,08 E au-delà. Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées et qui affiche la facture correspondante.

Exercice 4.5

Les habitants de Zorglub paient l'impôt selon les règles suivantes :

- les hommes de plus de 20 ans paient l'impôt
- les femmes paient l'impôt si elles ont entre 18 et 35 ans
- les autres ne paient pas d'impôt

Le programme demandera donc l'âge et le sexe du Zorglubien, et se prononcera donc ensuite sur le fait que l'habitant est imposable.

Exercice 4.6

Les élections législatives, en Guignolerie Septentrionale, obéissent à la règle suivante :

- lorsque l'un des candidats obtient plus de 50% des suffrages, il est élu dès le premier tour.
- en cas de deuxième tour, peuvent participer uniquement les candidats ayant obtenu au moins 12,5% des voix au premier tour.

Vous devez écrire un algorithme qui permette la saisie des scores de quatre candidats au premier tour. Cet algorithme traitera ensuite le candidat numéro 1 (et **uniquement** lui) : il dira s'il est élu, battu, s'il se trouve en ballottage favorable (il participe au second tour en étant arrivé en tête à l'issue du premier tour) ou défavorable (il participe au second tour sans avoir été en tête au premier tour).

Exercice 4.7

Une compagnie d'assurance automobile propose à ses clients quatre familles de tarifs identifiables par une couleur, du moins au plus onéreux : tarifs bleu, vert, orange et rouge. Le tarif dépend de la situation du conducteur :

- un conducteur de moins de 25 ans et titulaire du permis depuis moins de deux ans, se voit attribuer le tarif rouge, si toutefois il n'a jamais été responsable d'accident. Sinon, la compagnie refuse de l'assurer.
- un conducteur de moins de 25 ans et titulaire du permis depuis plus de deux ans, ou de plus de 25 ans mais titulaire du permis depuis moins de deux ans a le droit au tarif orange s'il n'a jamais provoqué d'accident, au tarif rouge pour un accident, sinon il est refusé.
- un conducteur de plus de 25 ans titulaire du permis depuis plus de deux ans bénéficie du tarif vert s'il n'est à l'origine d'aucun accident et du tarif orange pour un accident, du tarif rouge pour deux accidents, et refusé au-delà
- De plus, pour encourager la fidélité des clients acceptés, la compagnie propose un contrat de la couleur immédiatement la plus avantageuse s'il est entré dans la maison depuis plus d'un an.

Ecrire l'algorithme permettant de saisir les données nécessaires (sans contrôle de saisie) et de traiter ce problème. Avant de se lancer à corps perdu dans cet exercice, on pourra réfléchir un peu et s'apercevoir qu'il est plus simple qu'il n'en a l'air (cela s'appelle faire une analyse !)

Exercice 4.8

Ecrivez un algorithme qui a près avoir demandé un numéro de jour, de mois et d'année à l'utilisateur, renvoie s'il s'agit ou non d'une date valide.
Cet exercice est certes d'un manque d'originalité affligeant, mais après tout, en algorithmique comme ailleurs, il faut connaître ses classiques ! Et quand on a fait cela une fois dans sa vie, on apprécie pleinement l'existence d'un type numérique « date » dans certains langages…).
Il n'est sans doute pas inutile de rappeler rapidement que le mois de février compte 28 jours, sauf si l'année est bissextile, auquel cas il en compte 29. L'année est bissextile si elle est divisible par quatre. Toutefois, les années divisibles par 100 ne sont pas bissextiles, mais les années divisibles par 400 le sont. Ouf !
Un dernier petit détail : vous ne savez pas, pour l'instant, exprimer correctement en pseudo-code l'idée qu'un nombre A est divisible par un nombre B. Aussi, vous vous contenterez d'écrire en bons télégraphistes que A divisible par B se dit « A dp B ».

Corrigés des Exercices

Exercice 4.1

Aucune difficulté, il suffit d'appliquer la règle de la transformation du OU en ET vue en cours (loi de Morgan). Attention toutefois à la rigueur dans la transformation des conditions en leur contraire…

```
Si Tutu <= Toto + 4 ET Tata <> "OK" Alors
  Tutu ← Tutu - 1
Sinon
  Tutu ← Tutu + 1
Finsi
```

Exercice 4.2

```
Variables h, m en Numérique
Début
Ecrire "Entrez les heures, puis les minutes : "
Lire h, m
m ← m + 1
Si m = 60 Alors
  m ← 0
  h ← h + 1
FinSi
Si h = 24 Alors
  h ← 0
FinSi
Ecrire "Dans une minute il sera ", h, "heure(s) ", m, "minute(s)"
Fin
```

Exercice 4.3

```
Variables h, m, s en Numérique
Début
Ecrire "Entrez les heures, puis les minutes, puis les secondes : "
Lire h, m, s
s ← s + 1
Si s = 60 Alors
  s ← 0
  m ← m + 1
FinSi
Si m = 60 Alors
  m ← 0
  h ← h + 1
FinSi
Si h = 24 Alors
  h ← 0
FinSi
Ecrire "Dans une seconde il sera ", h, "h", m, "m et ", s, "s"
Fin
```

Exercice 4.4

```
Variables n, p en Numérique
Début
Ecrire "Nombre de photocopies : "
Lire n
Si n <= 10 Alors
  p ← n * 0,1
SinonSi n <= 30 Alors
  p ← 10 * 0,1 + (n – 10) * 0,09
Sinon
  p ← 10 * 0,1 + 20 * 0,09 + (n – 30) * 0,08
FinSi
Ecrire "Le prix total est: ", p
Fin
```

Exercice 4.5

```
Variable sex en Caractère
Variable age en Numérique
Variables C1, C2 en Booléen
Début
Ecrire "Entrez le sexe (M/F) : "
Lire sex
Ecrire "Entrez l'âge: "
Lire age
C1 ← sex = "M" ET age > 20
C2 ← sex = "F" ET (age > 18 ET age < 35)
Si C1 ou C2 Alors
  Ecrire "Imposable"
Sinon
  Ecrire "Non Imposable"
FinSi
Fin
```

Exercice 4.6

Cet exercice, du pur point de vue algorithmique, n'est pas très méchant. En revanche, il représente dignement la catégorie des énoncés piégés.
En effet, rien de plus facile que d'écrire : si le candidat a plus de 50%, il est élu, sinon s'il a plus de 12,5 %, il est au deuxième tour, sinon il est éliminé. Hé hé hé... mais il ne faut pas oublier que le candidat peut très bien avoir eu 20 % mais être tout de même éliminé, tout simplement parce que l'un des autres a fait plus de 50 % et donc qu'il n'y a pas de deuxième tour !...
Moralité : ne jamais se jeter sur la programmation avant d'avoir soigneusement mené l'analyse du problème à traiter.

```
Variables A, B, C, D en Numérique
Début
Ecrire "Entrez les scores des quatre prétendants :"
Lire A, B, C, D
C1 ← A > 50
C2 ← B > 50 ou C > 50 ou D > 50
C3 ← A >= B et A >= C et A >= D
C4 ← A >= 12,5
Si C1 Alors
  Ecrire "Elu au premier tour"
Sinonsi C2 ou Non(C4) Alors
  Ecrire "Battu, éliminé, sorti !!!"
SinonSi C3 Alors
  Ecrire "Ballotage favorable"
Sinon
  Ecrire "Ballotage défavorable"
FinSi
Fin
```

Exercice 4.7

Là encore, on illustre l'utilité d'une bonne analyse. Je propose deux corrigés différents. Le premier suit l'énoncé pas à pas. C'est juste, mais c'est vraiment lourd. La deuxième version s'appuie sur une vraie compréhension d'une situation pas si embrouillée qu'elle n'en a l'air.
Dans les deux cas, un recours aux variables booléennes aère sérieusement l'écriture.
Donc, premier corrigé, on suit le texte de l'énoncé pas à pas :

```
Variables age, perm, acc, assur en Numérique
Variables C1, C2, C3 en Booléen
Variable situ en Caractère
Début
Ecrire "Entrez l'âge: "
Lire age
Ecrire "Entrez le nombre d'années de permis: "
Lire perm
Ecrire "Entrez le nombre d'accidents: "
Lire acc
Ecrire "Entrez le nombre d'années d'assurance: "
Lire assur
C1 ← age >= 25
C2 ← perm >= 2
C3 ← assur > 1
Si Non(C1) et Non(C2) Alors
  Si acc = 0 Alors
    situ ← "Rouge"
  Sinon
    situ ← "Refusé"
  FinSi
Sinonsi ((Non(C1) et C2) ou (C1 et Non(C2)) Alors
  Si acc = 0 Alors
    situ ← "Orange"
  SinonSi acc = 1 Alors
    situ ← "Rouge"
  Sinon
    situ ← "Refusé"
  FinSi
Sinon
  Si acc = 0 Alors
    situ ← "Vert"
  SinonSi acc = 1 Alors
    situ ← "Orange"
  SinonSi acc = 2 Alors
    situ ← "Rouge"
  Sinon
    situ ← "Refusé"
  FinSi
FinSi
Si C3 Alors
  Si situ = "Rouge" Alors
    situ ← "Orange"
  SinonSi situ = "Orange" Alors
    situ ← "Orange"
  SinonSi situ = "Vert" Alors
    situ ← "Bleu"
  FinSi
FinSi
Ecrire "Votre situation : ", situ
Fin
```

Vous trouvez cela compliqué ? Oh, certes oui, ça l'est ! Et d'autant plus qu'en lisant entre les lignes, on pouvait s'apercevoir que ce galimatias de tarifs recouvre en fait une logique très simple : un système à points. Et il suffit de comptabiliser les points pour que tout s'éclaire... Reprenons juste après l'affectation des trois variables booléennes C1, C2, et C3. On écrit :

```
P ← 0
Si Non(C1) Alors
  P ← P + 1
FinSi
Si Non(C2) Alors
  P ← P + 1
FinSi
P ← P + acc
Si P < 3 et C3 Alors
  P ← P - 1
FinSi
Si P = -1 Alors
  situ ← "Bleu"
SinonSi P = 0 Alors
  situ ← "Vert"
SinonSi P = 1 Alors
  situ ← "Orange"
SinonSi P = 2 Alors
  situ ← "Rouge"
Sinon
  situ ← "Refusé"
FinSi
Ecrire "Votre situation : ", situ
Fin
```

Cool, non ?

Exercice 4.8

En ce qui concerne le début de cet algorithme, il n'y a aucune difficulté. C'est de la saisie bête et même pas méchante:

```
Variables J, M, A, JMax en Numérique
Variables VJ, VM, B en Booleen
Début
Ecrire "Entrez le numéro du jour"
Lire J
Ecrire "Entrez le numéro du mois"
Lire M
Ecrire "Entrez l'année"
Lire A
```

C'est évidemment ensuite que les ennuis commencent… La première manière d'aborder la chose consiste à se dire que fondamentalement, la structure logique de ce problème est très simple. Si nous créons deux variables booléennes VJ et VM, représentant respectivement la validité du jour et du mois entrés, la fin de l'algorithme sera d'une simplicité biblique (l'année est valide par définition, si on évacue le débat byzantin concernant l'existence de l'année zéro) :

```
Si VJ et VM alors
  Ecrire "La date est valide"
Sinon
  Ecrire "La date n'est pas valide"
FinSi
```

Toute la difficulté consiste à affecter correctement les variables VJ et VM, selon les valeurs des variables J, M et A. Dans l'absolu, VJ et VM pourraient être les objets d'une affectation monstrueuse, avec des conditions atrocement composées. Mais franchement, écrire ces conditions en une seule fois est un travail de bénédictin sans grand intérêt. Pour éviter d'en arriver à une telle extrémité, on peut sérier la difficulté en créant deux variables supplémentaires :

B : variable booléenne qui indique s'il s'agit d'une année bissextile
JMax : variable numérique qui indiquera le dernier jour valable pour le mois entré.

Avec tout cela, on peut y aller et en ressortir vivant.
On commence par initialiser nos variables booléennes, puis on traite les années, puis les mois, puis les jours.
On note "dp" la condition "divisible par" :

```
B ← A dp 400 ou (non(A dp 100) et A dp 4)
Jmax ← 0
VM ← M >= 1 et M =< 12
Si VM Alors
  Si M = 2 et B Alors
    JMax ← 29
  SinonSi M = 2 Alors
    JMax ← 28
  SinonSi M = 4 ou M = 6 ou M = 9 ou M = 11 Alors
    JMax ← 30
  Sinon
    JMax ← 31
  FinSi
  VJ ← J >= 1 et J =< Jmax
FinSi
```

Cette solution a le mérite de ne pas trop compliquer la structure des tests, et notamment de ne pas répéter l'écriture finale à l'écran. Les variables booléennes intermédiaires nous épargnent des conditions composées trop lourdes, mais celles-ci restent néanmoins sérieuses.

Une approche différente consisterait à limiter les conditions composées, quitte à le payer par une structure beaucoup plus exigeante de tests imbriqués. Là encore, on évite de jouer les extrémistes et l'on s'autorise quelques conditions composées lorsque cela nous simplifie l'existence. On pourrait aussi dire que la solution précédente "part de la fin" du problème (la date est elle valide ou non ?), alors que celle qui suit "part du début" (quelles sont les données entrées au clavier ?) :

```
Si M < 1 ou M > 12 Alors
  Ecrire "Date Invalide"
SinonSi M = 2 Alors
  Si A dp 400 Alors
    Si J < 1 ou J > 29 Alors
      Ecrire "Date Invalide"
    Sinon
      Ecrire "Date Valide"
    FinSi
  SinonSi A dp 100 Alors
    Si J < 1 ou J > 28 Alors
      Ecrire "Date Invalide"
    Sinon
      Ecrire "Date Valide"
    FinSi
  SinonSi A dp 4 Alors
    Si J < 1 ou J > 28 Alors
      Ecrire "Date Invalide"
    Sinon
      Ecrire "Date Valide"
    FinSi
  Sinon
    Si J < 1 ou J > 28 Alors
```

```
      Ecrire "Date Invalide"
    Sinon
      Ecrire "Date Valide"
    FinSi
  FinSi
SinonSi M = 4 ou M = 6 ou M = 9 ou M = 11 Alors
  Si J < 1 ou J > 30 Alors
    Ecrire "Date Invalide"
  Sinon
    Ecrire "Date Valide"
  FinSi
Sinon
  Si J < 1 ou J > 31 Alors
    Ecrire "Date Invalide"
  Sinon
    Ecrire "Date Valide"
  FinSi
FinSi
```

On voit que dans ce cas, l'alternative finale (Date valide ou invalide) se trouve répétée un grand nombre de fois. Ce n'est en soi ni une bonne, ni une mauvaise chose. C'est simplement une question de choix stylistique.

Personnellement, j'avoue préférer assez nettement la première solution, qui fait ressortir beaucoup plus clairement la structure logique du problème (il n'y a qu'une seule alternative, autant que cette alternative ne soit écrite qu'une seule fois).

Il convient enfin de citer une solution très simple et élégante, un peu plus difficile peut-être à imaginer du premier coup, mais qui avec le recul apparaît comme très immédiate. Sur le fond, cela consiste à dire qu'il y a quatre cas pour qu'une date soit valide : celui d'un jour compris entre 1 et 31 dans un mois à 31 jours, celui d'un jour compris entre 1 et 30 dans un mois à 30 jours, celui d'un jour compris entre 1 et 29 en février d'une année bissextile, et celui d'un jour de février compris entre 1 et 28. Ainsi :

```
B ← (A dp 4 et Non(A dp 100)) ou A dp 400
K1 ← (m=1 ou m=3 ou m=5 ou m=7 ou m=8 ou m=10 ou m=12) et (J>=1 et J=<31)
K2 ← (m=4 ou m=6 ou m=9 ou m=11) et (J>=1 et J=<30)
K3 ← m=2 et B et J>=1 et J=<29
K4 ← m=2 et J>=1 et J=<28
Si K1 ou K2 ou K3 ou K4 Alors
  Ecrire "Date valide"
Sinon
  Ecrire "Date non valide"
FinSi
Fin
```

PARTIE 5
Enonce des Exercices

Exercice 5.1

Ecrire un algorithme qui demande à l'utilisateur un nombre compris entre 1 et 3 jusqu'à ce que la réponse convienne.

Exercice 5.2

Ecrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : « Plus petit ! », et inversement, « Plus grand ! » si le nombre est inférieur à 10.

Exercice 5.3

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.

Exercice 5.4

Ecrire un algorithme qui demande un nombre de départ, et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :

```
Table de 7 :
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
…
7 x 10 = 70
```

Exercice 5.5

Ecrire un algorithme qui demande un nombre de départ, et qui calcule la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :

```
1 + 2 + 3 + 4 + 5 = 15
```

NB : on souhaite afficher uniquement le résultat, pas la décomposition du calcul.

Exercice 5.6

Ecrire un algorithme qui demande un nombre de départ, et qui calcule sa factorielle.

NB : la factorielle de 8, notée 8 !, vaut

1 x 2 x 3 x 4 x 5 x 6 x 7 x 8

Exercice 5.7

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 20 nombres :

```
Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
etc.
```

```
Entrez le nombre numéro 20 : 6
Le plus grand de ces nombres est  : 14
```

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre :

```
C'était le nombre numéro 2
```

Exercice 5.8

Réécrire l'algorithme précédent, mais cette fois-ci on ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un zéro.

Exercice 5.9

Lire la suite des prix (en euros entiers et terminée par zéro) des achats d'un client. Calculer la somme qu'il doit, lire la somme qu'il paye, et simuler la remise de la monnaie en affichant les textes "10 Euros", "5 Euros" et "1 Euro" autant de fois qu'il y a de coupures de chaque sorte à rendre.

Exercice 5.10

Écrire un algorithme qui permette de connaître ses chances de gagner au tiercé, quarté, quinté et autres impôts volontaires.

On demande à l'utilisateur le nombre de chevaux partants, et le nombre de chevaux joués. Les deux messages affichés devront être :

```
Dans l'ordre : une chance sur X de gagner
Dans le désordre : une chance sur Y de gagner
```

X et Y nous sont donnés par la formule suivante, si n est le nombre de chevaux partants et p le nombre de chevaux joués (on rappelle que le signe ! signifie "factorielle", comme dans l'exercice 5.6 ci-dessus) :

```
X = n ! / (n - p) !
Y = n ! / (p ! * (n – p) !)
```

NB : cet algorithme peut être écrit d'une manière simple, mais relativement peu performante. Ses performances peuvent être singulièrement augmentées par une petite astuce. Vous commencerez par écrire la manière la plus simple, puis vous identifierez le problème, et écrirez une deuxième version permettant de le résoudre.

Corrigés des Exercices

Exercice 5.1

```
Variable N en Entier
Debut
N ← 0
Ecrire "Entrez un nombre entre 1 et 3"
TantQue N < 1 ou N > 3
```

```
  Lire N
    Si N < 1 ou N > 3 Alors
      Ecrire "Saisie erronée. Recommencez”
    FinSi
  FinTantQue
Fin
```

Exercice 5.2

```
Variable N en Entier
Debut
N ← 0
Ecrire "Entrez un nombre entre 10 et 20"
TantQue N < 10 ou N > 20
  Lire N
  Si N < 10 Alors
    Ecrire "Plus grand !"
  SinonSi N > 20 Alors
    Ecrire "Plus petit !"
  FinSi
FinTantQue
Fin
```

Exercice 5.3

```
Variables N, i en Entier
Debut
Ecrire "Entrez un nombre : "
Lire N
Ecrire "Les 10 nombres suivants sont : "
Pour i ← N + 1 à N + 10
  Ecrire i
i Suivant
Fin
```

Exercice 5.4

```
Variables N, i en Entier
Debut
Ecrire "Entrez un nombre : "
Lire N
Ecrire "La table de multiplication de ce nombre est : "
Pour i ← 1 à 10
  Ecrire N, " x ", i, " = ", n*i
i Suivant
Fin
```

Exercice 5.5

```
Variables N, i, Som en Entier
Debut
Ecrire "Entrez un nombre : "
Lire N
Som ← 0
Pour i ← 1 à N
  Som ← Som + i
i Suivant
Ecrire "La somme est : ", Som
Fin
```

Exercice 5.6

```
Variables N, i, F en Entier
Debut
Ecrire "Entrez un nombre : "
Lire N
F ← 1
Pour i ← 2 à N
  F ← F * i
i Suivant
Ecrire "La factorielle est : ", F
Fin
```

Exercice 5.7

```
Variables N, i, PG en Entier
Debut
PG ← 0
Pour i ← 1 à 20
  Ecrire "Entrez un nombre : "
  Lire N
  Si i = 1 ou N > PG Alors
    PG ← N
  FinSi
i Suivant
Ecrire "Le nombre le plus grand était : ", PG
Fin
```

En ligne 3, on peut mettre n'importe quoi dans PG, il suffit que cette variable soit affectée pour que le premier passage en ligne 7 ne provoque pas d'erreur.

Pour la version améliorée, cela donne :

```
Variables N, i, PG, IPG en Entier
Debut
PG ← 0
Pour i ← 1 à 20
  Ecrire "Entrez un nombre : "
  Lire N
  Si i = 1 ou N > PG Alors
    PG ← N
    IPG ← i
  FinSi
i Suivant
Ecrire "Le nombre le plus grand était : ", PG
Ecrire "Il a été saisi en position numéro ", IPG
Fin
```

Exercice 5.8

```
Variables N, i, PG, IPG en Entier
Debut
N ← 1
i ← 0
PG ← 0
TantQue N <> 0
  Ecrire "Entrez un nombre : "
  Lire N
  i ← i + 1
```

```
  Si i = 1 ou N > PG Alors
    PG ← N
    IPG ← i
  FinSi
FinTantQue
Ecrire "Le nombre le plus grand était : ", PG
Ecrire "Il a été saisi en position numéro ", IPG
Fin
```

Exercice 5.9

```
Variables FF, somdue, M, IPG, Reste, Nb10F, Nb5F En Entier
Debut
E ← 1
somdue ← 0
TantQue E <> 0
  Ecrire "Entrez le montant : "
  Lire E
  somdue ← somdue + E
FinTantQue
Ecrire "Vous devez :", E, " euros"
Ecrire "Montant versé :"
Lire M
Reste ← M - E
Nb10E ← 0
TantQue Reste >= 10
  Nb10E ← Nb10E + 1
  Reste ← Reste – 10
FinTantQue
Nb5E ← 0
Si Reste >= 5
  Nb5E ← 1
  Reste ← Reste – 5
FinSi
Ecrire "Rendu de la monnaie :"
Ecrire "Billets de 10 E : ", Nb10E
Ecrire "Billets de  5 E : ", Nb5E
Ecrire "Pièces de 1 E : ", reste
Fin
```

Exercice 5.10

Spontanément, on est tenté d'écrire l'algorithme suivant :

```
Variables N, P, i, Numé, Déno1, Déno2 en Entier
Debut Ecrire "Entrez le nombre de chevaux partants : "
Lire N
Ecrire "Entrez le nombre de chevaux joués : "
Lire P
Numé ← 1
Pour i ← 2 à N
  Numé ← Numé * i
i Suivant
Déno1 ← 1
Pour i ← 2 à N-P
  Déno1 ← Déno1 * i
i Suivant
Déno2 ← 1
```

```
Pour i ← 2 à P
  Déno2 ← Déno2 * i
i Suivant
Ecrire "Dans l'ordre, une chance sur ", Numé / Déno1
Ecrire "Dans le désordre, une sur ", Numé / (Déno1 * Déno2)
Fin
```

Cette version, formellement juste, comporte tout de même deux faiblesses.

La première, et la plus grave, concerne la manière dont elle calcule le résultat final. Celui-ci est le quotient d'un nombre par un autre ; or, ces nombres auront rapidement tendance à être très grands. En calculant, comme on le fait ici, d'abord le numérateur, puis ensuite le dénominateur, on prend le risque de demander à la machine de stocker des nombres trop grands pour qu'elle soit capable de les coder (cf. le préambule). C'est d'autant plus bête que rien ne nous oblige à procéder ainsi : on n'est pas obligé de passer par la division de deux très grands nombres pour obtenir le résultat voulu.

La deuxième remarque est qu'on a programmé ici trois boucles successives. Or, en y regardant bien, on peut voir qu'après simplification de la formule, ces trois boucles comportent le même nombre de tours ! (si vous ne me croyez pas, écrivez un exemple de calcul et biffez les nombres identiques au numérateur et au dénominateur). Ce triple calcul (ces trois boucles) peut donc être ramené(es) à un(e) seul(e). Et voilà le travail, qui est non seulement bien plus court, mais aussi plus performant :

```
Variables N, P, i, O, F en Entier
Debut
Ecrire "Entrez le nombre de chevaux partants : "
Lire N
Ecrire "Entrez le nombre de chevaux joués : "
Lire P
A ← 1
B ← 1
Pour i ← 1 à P
  A ← A * (i + N - P)
  B ← B * i
i Suivant
Ecrire "Dans l'ordre, une chance sur ", A
Ecrire "Dans le désordre, une chance sur ", A / B
Fin
```

PARTIE 6
Enonce des Exercices

Exercice 6.1

Ecrire un algorithme qui déclare et remplisse un tableau de 7 valeurs numériques en les mettant toutes à zéro.

Exercice 6.2

Ecrire un algorithme qui déclare et remplisse un tableau contenant les six voyelles de l'alphabet latin.

Exercice 6.3

Ecrire un algorithme qui déclare un tableau de 9 notes, dont on fait ensuite saisir les valeurs par l'utilisateur.

Exercice 6.4

Que produit l'algorithme suivant ?

```
Tableau Nb(5) en Entier
Variable i en Entier
Début
Pour i ← 0 à 5
  Nb(i) ← i * i
i suivant
Pour i ← 0 à 5
  Ecrire Nb(i)
i suivant
Fin
```

Peut-on simplifier cet algorithme avec le même résultat ?

Exercice 6.5

Que produit l'algorithme suivant ?

```
Tableau N(6) en Entier
Variables i, k en Entier
Début
N(0) ← 1
Pour k ← 1 à 6
  N(k) ← N(k-1) + 2
k Suivant
Pour i ← 0 à 6
  Ecrire N(i)
i suivant
Fin
```

Peut-on simplifier cet algorithme avec le même résultat ?

Exercice 6.6

Que produit l'algorithme suivant ?

```
Tableau Suite(7) en Entier
Variable i en Entier
Début
Suite(0) ← 1
Suite(1) ← 1
```

```
Pour i ← 2 à 7
  Suite(i) ← Suite(i-1) + Suite(i-2)
i suivant
Pour i ← 0 à 7
  Ecrire Suite(i)
i suivant
Fin
```

Exercice 6.7

Ecrivez la fin de l'algorithme 6.3 afin que le calcul de la moyenne des notes soit effectué et affiché à l'écran.

Exercice 6.8

Ecrivez un algorithme permettant à l'utilisateur de saisir un nombre quelconque de valeurs, qui devront être stockées dans un tableau. L'utilisateur doit donc commencer par entrer le nombre de valeurs qu'il compte saisir. Il effectuera ensuite cette saisie. Enfin, une fois la saisie terminée, le programme affichera le nombre de valeurs négatives et le nombre de valeurs positives.

Exercice 6.9

Ecrivez un algorithme calculant la somme des valeurs d'un tableau (on suppose que le tableau a été préalablement saisi).

Exercice 6.10

Ecrivez un algorithme constituant un tableau, à partir de deux tableaux de même longueur préalablement saisis. Le nouveau tableau sera la somme des éléments des deux tableaux de départ.

Tableau 1 :

4	8	7	9	1	5	4	6

Tableau 2 :

7	6	5	2	1	3	7	4

Tableau à constituer :

11	14	12	11	2	8	11	10

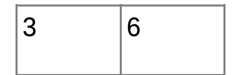
Exercice 6.11

Toujours à partir de deux tableaux précédemment saisis, écrivez un algorithme qui calcule le schtroumpf des deux tableaux. Pour calculer le schtroumpf, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout. Par exemple si l'on a :

Tableau 1 :

4	8	7	12

Tableau 2 :



3	6

Le Schtroumpf sera :

3 * 4 + 3 * 8 + 3 * 7 + 3 * 12 + 6 * 4 + 6 * 8 + 6 * 7 + 6 * 12 = 279

Exercice 6.12

Ecrivez un algorithme qui permette la saisie d'un nombre quelconque de valeurs, sur le principe de l'ex 6.8. Toutes les valeurs doivent être ensuite augmentées de 1, et le nouveau tableau sera affiché à l'écran.

Exercice 6.13

Ecrivez un algorithme permettant, toujours sur le même principe, à l'utilisateur de saisir un nombre déterminé de valeurs. Le programme, une fois la saisie terminée, renvoie la plus grande valeur en précisant quelle position elle occupe dans le tableau. On prendra soin d'effectuer la saisie dans un premier temps, et la recherche de la plus grande valeur du tableau dans un second temps.

Exercice 6.14

Toujours et encore sur le même principe, écrivez un algorithme permettant, à l'utilisateur de saisir les notes d'une classe. Le programme, une fois la saisie terminée, renvoie le nombre de ces notes supérieures à la moyenne **de la classe**.

Corrigés des Exercices

Exercice 6.1

```
Tableau Truc(6) en Numérique
Variable i en Numérique
Debut
Pour i ← 0 à 6
  Truc(i) ← 0
```

```
i Suivant
Fin
```

Exercice 6.2

```
Tableau Truc(5) en Caractère
Debut
Truc(0) ← "a"
Truc(1) ← "e"
Truc(2) ← "i"
Truc(3) ← "o"
Truc(4) ← "u"
Truc(5) ← "y"
Fin
```

Exercice 6.3

```
Tableau Notes(8) en Numérique
Variable i en Numérique
Pour i ← 0 à 8
  Ecrire "Entrez la note numéro ", i + 1
  Lire Notes(i)
i Suivant
Fin
```

Exercice 6.4

Cet algorithme remplit un tableau avec six valeurs : 0, 1, 4, 9, 16, 25.
Il les écrit ensuite à l'écran. Simplification :

```
Tableau Nb(5) en Numérique
Variable i en Numérique
Début
Pour i ← 0 à 5
  Nb(i) ← i * i
  Ecrire Nb(i)
i Suivant
Fin
```

Exercice 6.5

Cet algorithme remplit un tableau avec les sept valeurs : 1, 3, 5, 7, 9, 11, 13.
Il les écrit ensuite à l'écran. Simplification :

```
Tableau N(6) en Numérique
Variables i, k en Numérique
Début
N(0) ← 1
Ecrire N(0)
Pour k ← 1 à 6
  N(k) ← N(k-1) + 2
  Ecrire N(k)
k Suivant
Fin
```

Exercice 6.6

Cet algorithme remplit un tableau de 8 valeurs : 1, 1, 2, 3, 5, 8, 13, 21

Exercice 6.7

```
Variable S en Numérique
Tableau Notes(8) en Numérique
```

```
Debut
s ← 0
Pour i ← 0 à 8
  Ecrire "Entrez la note n° ", i + 1
  Lire Notes(i)
  s ← s + Notes(i)
i Suivant
Ecrire "Moyenne :", s/9
Fin
```

Exercice 6.8

```
Variables Nb, Nbpos, Nbneg en Numérique
Tableau T() en Numérique
Debut
Ecrire "Entrez le nombre de valeurs :"
Lire Nb
Redim T(Nb-1)
Nbpos ← 0
Nbneg ← 0
Pour i ← 0 à Nb - 1
  Ecrire "Entrez le nombre n° ", i + 1
  Lire T(i)
  Si T(i) > 0 alors
    Nbpos ← Nbpos + 1
  Sinon
    Nbneg ← Nbneg + 1
  Finsi
i Suivant
Ecrire "Nombre de valeurs positives : ", Nbpos
Ecrire "Nombre de valeurs négatives : ", Nbneg
Fin
```

Exercice 6.9

```
Variables i, Som, N en Numérique
Tableau T() en Numérique
Debut
… (on ne programme pas la saisie du tableau, dont on suppose qu'il compte N éléments)
Redim T(N-1)
…
Som ← 0
Pour i ← 0 à N - 1
  Som ← Som + T(i)
i Suivant
Ecrire "Somme des éléments du tableau : ", Som
Fin
```

Exercice 6.10

```
Variables i, N en Numérique
Tableaux T1(), T2(), T3() en Numérique
Debut
… (on suppose que T1 et T2 comptent N éléments, et qu'ils sont déjà saisis)
Redim T3(N-1)
…
Pour i ← 0 à N - 1
  T3(i) ← T1(i) + T2(i)
```

```
i Suivant
Fin
```

Exercice 6.11

```
Variables i, j, N1, N2, S en Numérique
Tableaux T1(), T2() en Numérique
Debut
```

… On ne programme pas la saisie des tableaux T1 et T2.
On suppose que T1 possède N1 éléments, et que T2 en possède T2)
…

```
S ← 0
Pour i ← 0 à N1 – 1
  Pour j ← 0 à N2 – 1
    S ← S + T1(i) * T2(j)
  j Suivant
i Suivant
Ecrire "Le schtroumpf est : ", S
Fin
```

Exercice 6.12

```
Variables Nb, i en Numérique
Tableau T() en Numérique
Debut
Ecrire "Entrez le nombre de valeurs : "
Lire Nb
Redim T(Nb-1)
Pour i ← 0 à Nb - 1
  Ecrire "Entrez le nombre n° ", i + 1
  Lire T(i)
i Suivant
Ecrire "Nouveau tableau : "
Pour i ← 0 à Nb – 1
  T(i) ← T(i) + 1
  Ecrire T(i)
i Suivant
Fin
```


Exercice 6.13

```
Variables Nb, Posmaxi en Numérique
Tableau T() en Numérique
Ecrire "Entrez le nombre de valeurs :"
Lire Nb
Redim T(Nb-1)
Pour i ← 0 à Nb - 1
  Ecrire "Entrez le nombre n° ", i + 1
  Lire T(i)
i Suivant
Posmaxi ← 0
Pour i ← 0 à Nb - 1
  Si T(i) > T(Posmaxi) alors
    Posmaxi ← i
  Finsi
i Suivant
Ecrire "Element le plus grand : ", T(Posmaxi)
```

```
Ecrire "Position de cet élément : ", Posmaxi
Fin
```

Exercice 6.14

```
Variables Nb, i, Som, Moy, Nbsup en Numérique
Tableau T() en Numérique
Debut
Ecrire "Entrez le nombre de notes à saisir : "
Lire Nb
Redim T(Nb-1)
Pour i ← 0 à Nb - 1
  Ecrire "Entrez le nombre n° ", i + 1
  Lire T(i)
i Suivant
Som ← 0
Pour i ← 0 à Nb - 1
  Som ← Som + T(i)
i Suivant
Moy ← Som / Nb
NbSup ← 0
Pour i ← 0 à Nb - 1
  Si T(i) > Moy Alors
    NbSup ← NbSup + 1
  FinSi
i Suivant
Ecrire NbSup, " élèves dépassent la moyenne de la classe"
Fin
```

PARTIE 7
Enonce des Exercices

Exercice 7.1

Ecrivez un algorithme qui permette de saisir un nombre quelconque de valeurs, et qui les range au fur et à mesure dans un tableau. Le programme, une fois la saisie terminée, doit dire si les éléments du tableau sont tous consécutifs ou non.

Par exemple, si le tableau est :

12	13	14	15	16	17	18

ses éléments sont tous consécutifs. En revanche, si le tableau est :

9	10	11	15	16	17	18

ses éléments ne sont pas tous consécutifs.

Exercice 7.2

Ecrivez un algorithme qui trie un tableau dans l'ordre décroissant.

Vous écrirez bien entendu deux versions de cet algorithme, l'une employant le tri par insertion, l'autre le tri à bulles.

Exercice 7.3

Ecrivez un algorithme qui inverse l'ordre des éléments d'un tableau dont on suppose qu'il a été préalablement saisi (« les premiers seront les derniers… »)

Exercice 7.4

Ecrivez un algorithme qui permette à l'utilisateur de supprimer une valeur d'un tableau préalablement saisi. L'utilisateur donnera l'indice de la valeur qu'il souhaite supprimer. Attention, il ne s'agit pas de remettre une valeur à zéro, mais bel et bien de la supprimer du tableau lui-même ! Si le tableau de départ était :

12	8	4	45	64	9	2

Et que l'utilisateur souhaite supprimer la valeur d'indice 4, le nouveau tableau sera :

12	8	4	45	9	2

Exercice 7.5

Ecrivez l'algorithme qui recherche un mot saisi au clavier dans un dictionnaire. Le dictionnaire est supposé être codé dans un tableau préalablement rempli et trié.

Corrigés des Exercices

Exercice 7.1

```
Variables Nb, i en Entier
Variable Flag en Booleen
Tableau T() en Entier
Debut
Ecrire "Entrez le nombre de valeurs :"
Lire Nb
Redim T(Nb-1)
Pour i ← 0 à Nb - 1
  Ecrire "Entrez le nombre n° ", i + 1
  Lire T(i)
i Suivant
Flag ← Vrai
Pour i ← 1 à Nb - 1
  Si T(i) <> T(i – 1) + 1 Alors
    Flag ← Faux
  FinSi
i Suivant
Si Flag Alors
  Ecrire "Les nombres sont consécutifs"
Sinon
  Ecrire "Les nombres ne sont pas consécutifs"
FinSi
Fin
```

Cette programmation est sans doute la plus spontanée, mais elle présente le défaut d'examiner la totalité du tableau, même lorsqu'on découvre dès le départ deux éléments non consécutifs. Aussi, dans le cas d'un grand tableau, est-elle dispendieuse en temps de traitement. Une autre manière de procéder serait de sortir de la boucle dès que deux éléments non consécutifs sont détectés. La deuxième partie de l'algorithme deviendrait donc :

```
i ← 1
TantQue T(i) = T(i – 1) + 1 et i < Nb - 1
  i ← i + 1
FinTantQue
Si T(i) = T(i – 1) + 1 Alors
  Ecrire "Les nombres sont consécutifs"
Sinon
  Ecrire "Les nombres ne sont pas consécutifs"
FinSi
```

Exercice 7.2

On suppose que N est le nombre d’éléments du tableau. Tri par insertion :

```
…
Pour i ← 0 à N - 2
  posmaxi = i
  Pour j ← i + 1 à N - 1
    Si t(j) > t(posmaxi) alors
      posmaxi ← j
    Finsi
  j suivant
  temp ← t(posmaxi)
  t(posmaxi) ← t(i)
  t(i) ← temp
i suivant
Fin
```

Tri à bulles :

```
…
Yapermut ← Vrai
TantQue Yapermut
  Yapermut ← Faux
  Pour i ← 0 à N - 2
    Si t(i) < t(i + 1) Alors
      temp ← t(i)
      t(i) ← t(i + 1)
      t(i + 1) ← temp
      Yapermut ← Vrai
    Finsi
  i suivant
FinTantQue
Fin
```

Exercice 7.3

On suppose que n est le nombre d'éléments du tableau préalablement saisi

```
…
Pour i ← 0 à (N-1)/2
  Temp ← T(i)
  T(i) ← T(N-1-i)
  T(N-1-i) ← Temp
i suivant
Fin
```

Exercice 7.4

```
…
Ecrire "Rang de la valeur à supprimer ?"
Lire S
Pour i ← S à N-2
  T(i) ← T(i+1)
i suivant
Redim T(N–1)
Fin
```

Exercice 7.5

N est le nombre d'éléments du tableau Dico(), contenant les mots du dictionnaire, tableau préalablement rempli.

```
Variables Sup, Inf, Comp en Entier
Variables Fini en Booléen
Début
Ecrire "Entrez le mot à vérifier"
Lire Mot
```

On définit les bornes de la partie du tableau à considérer

```
Sup ← N - 1
Inf ← 0
Fini ← Faux
TantQue Non Fini
```

Comp désigne l'indice de l'élément à comparer. En bonne rigueur, il faudra veiller à ce que Comp soit bien un nombre entier, ce qui pourra s'effectuer de différentes manières selon les langages.

```
  Comp ← (Sup + Inf)/2
```

Si le mot se situe avant le point de comparaison, alors la borne supérieure change, la borne inférieure ne bouge pas.

```
  Si Mot < Dico(Comp) Alors
    Sup ← Comp - 1
```

Sinon, c'est l'inverse

```
  Sinon
    Inf ← Comp + 1
  FinSi
  Fini ← Mot = Dico(Comp) ou Sup < Inf
FinTantQue
Si Mot = Dico(Comp) Alors
  Ecrire "le mot existe"
Sinon
  Ecrire "Il n'existe pas"
Finsi
Fin
```



PARTIE 8
Enonce des Exercices

Exercice 8.1

Écrivez un algorithme remplissant un tableau de 6 sur 13, avec des zéros.

Exercice 8.2

Quel résultat produira cet algorithme ?

```
Tableau X(1, 2) en Entier
Variables i, j, val en Entier
Début
Val ← 1
Pour i ← 0 à 1
  Pour j ← 0 à 2
    X(i, j) ← Val
    Val ← Val + 1
  j Suivant
i Suivant
Pour i ← 0 à 1
  Pour j ← 0 à 2
    Ecrire X(i, j)
  j Suivant
i Suivant
Fin
```

Exercice 8.3

Quel résultat produira cet algorithme ?

```
Tableau X(1, 2) en Entier
Variables i, j, val en Entier
Début
Val ← 1
Pour i ← 0 à 1
  Pour j ← 0 à 2
    X(i, j) ← Val
    Val ← Val + 1
  j Suivant
i Suivant
Pour j ← 0 à 2
  Pour i ← 0 à 1
    Ecrire X(i, j)
  i Suivant
j Suivant
Fin
```

Exercice 8.4

Quel résultat produira cet algorithme ?

```
Tableau T(3, 1) en Entier
Variables k, m, en Entier
Début
Pour k ← 0 à 3
  Pour m ← 0 à 1
    T(k, m) ← k + m
  m Suivant
k Suivant
Pour k ← 0 à 3
  Pour m ← 0 à 1
```

```
    Ecrire T(k, m)
  m Suivant
k Suivant
Fin
```

Exercice 8.5

Mêmes questions, en remplaçant la ligne :

```
T(k, m) ← k + m
```

par

```
T(k, m) ← 2 * k + (m + 1)
```

puis par :

```
T(k, m) ← (k + 1) + 4 * m
```

Exercice 8.6

Soit un tableau T à deux dimensions (12, 8) préalablement rempli de valeurs numériques.

Écrire un algorithme qui recherche la plus grande valeur au sein de ce tableau.

Exercice 8.7

Écrire un algorithme de jeu de dames très simplifié.

L'ordinateur demande à l'utilisateur dans quelle case se trouve son pion (quelle ligne, quelle colonne). On met en place un contrôle de saisie afin de vérifier la validité des valeurs entrées.

Ensuite, on demande à l'utilisateur quel mouvement il veut effectuer : 0 (en haut à gauche), 1 (en haut à droite), 2 (en bas à gauche), 3 (en bas à droite).

Si le mouvement est impossible (i.e. on sort du damier), on le signale à l'utilisateur et on s'arrête là . Sinon, on déplace le pion et on affiche le damier résultant, en affichant un « O » pour une case vide et un « X » pour la case où se trouve le pion.

Corrigés des Exercices

Exercice 8.1

```
Tableau Truc(5, 12) en Entier
Debut
Pour i ← 0 à 5
  Pour j ← 0 à 12
    Truc(i, j) ← 0
```

```
  j Suivant
i Suivant
Fin
```

Exercice 8.2

Cet algorithme remplit un tableau de la manière suivante:

```
X(0, 0) = 1
X(0, 1) = 2
X(0, 2) = 3
X(1, 0) = 4
X(1, 1) = 5
X(1, 2) = 6
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

Exercice 8.3

Cet algorithme remplit un tableau de la manière suivante:

```
X(0, 0) = 1
X(1, 0) = 4
X(0, 1) = 2
X(1, 1) = 5
X(0, 2) = 3
X(1, 2) = 6
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

Exercice 8.4

Cet algorithme remplit un tableau de la manière suivante:

```
T(0, 0) = 0
T(0, 1) = 1
T(1, 0) = 1
T(1, 1) = 2
T(2, 0) = 2
T(2, 1) = 3
T(3, 0) = 3
T(3, 1) = 4
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

Exercice 8.5

Version a : cet algorithme remplit un tableau de la manière suivante:

```
T(0, 0) = 1
T(0, 1) = 2
T(1, 0) = 3
T(1, 1) = 4
T(2, 0) = 5
T(2, 1) = 6
T(3, 0) = 7
T(3, 1) = 8
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

Version b : cet algorithme remplit un tableau de la manière suivante:

```
T(0, 0) = 1
T(0, 1) = 5
T(1, 0) = 2
T(1, 1) = 6
T(2, 0) = 3
```

```
T(2, 1) = 7
T(3, 0) = 4
T(3, 1) = 8
```

Il écrit ensuite ces valeurs à l'écran, dans cet ordre.

Exercice 8.6

```
Variables i, j, iMax, jMax en Numérique
Tableau T(12, 8) en Numérique
```

Le principe de la recherche dans un tableau à deux dimensions est strictement le même que dans un tableau à une dimension, ce qui ne doit pas nous étonner. La seule chose qui change, c'est qu'ici le balayage requiert deux boucles imbriquées, au lieu d'une seule.

```
Debut
...
iMax ← 0
jMax ← 0
Pour i ← 0 à 12
  Pour j ← 0 à 8
    Si T(i,j) > T(iMax,jMax) Alors
      iMax ← i
      jMax ← j
    FinSi
  j Suivant
i Suivant
Ecrire "Le plus grand élément est ", T(iMax, jMax)
Ecrire "Il se trouve aux indices ", iMax, "; ", jMax
Fin
```

Exercice 8.7

```
Variables i, j , posi, posj, i2, j2 en Entier
Variables Correct, MoveOK en Booléen
Tableau Damier(7, 7) en Booléen
Tableau Mouv(3, 1) en Entier
```

Le damier contenant un seul pion, on choisit de le coder à l'économie, en le représentant par un tableau de booléens à deux dimensions. Dans chacun des emplacements de ce damier, Faux signifie l'absence du pion, Vrai sa présence.

Par ailleurs, on emploie une méchante astuce, pas obligatoire, mais bien pratique dans beaucoup de situations. L'idée est de faire correspondre les choix possibles de l'utilisateur avec les mouvements du pion. On entre donc dans un tableau Mouv à deux dimensions, les déplacements du pion selon les quatre directions, en prenant soin que chaque ligne du tableau corresponde à une saisie de l'utilisateur. La première valeur étant le déplacement en i, la seconde le déplacement en j. Ceci nous épargnera par la suite de faire quatre fois les mêmes tests.

```
Debut
```

Choix 0 : pion en haut à droite

```
Mouv(0, 0) ← -1
Mouv(0, 1) ← -1
```

Choix 1 : pion en haut à droite

```
Mouv(1, 0) ← -1
Mouv(1, 1) ← 1
```

Choix 2 : pion en bas à gauche

```
Mouv(2, 0) ← 1
Mouv(2, 1) ← -1
```

Choix 3 : pion en bas à droite

```
Mouv(3, 0) ← 1
Mouv(3, 1) ← 1
```

Initialisation du damier; le pion n'est pour le moment nulle part

```
Pour i ← 0 à 7
  Pour j ← 0 à 7
    Damier(i, j) ← Faux
  j suivant
i suivant
```

Saisie de la coordonnée en i ("posi") avec contrôle de saisie

```
Correct ← Faux
TantQue Non Correct
  Ecrire "Entrez la ligne de votre pion: "
  Lire posi
  Si posi >= 0 et posi <= 7 Alors
    Correct ← vrai
  Finsi
Fintantque
```

Saisie de la coordonnée en j ("posj") avec contrôle de saisie

```
Correct ← Faux
TantQue Non Correct
  Ecrire "Entrez la colonne de votre pion: "
  Lire posj
    Si posj >= 0 et posj <= 7 Alors
      Correct ← Vrai
    Finsi
Fintantque
```

Positionnement du pion sur le damier virtuel.

```
Damier(posi, posj) ← Vrai
```

Saisie du déplacement, avec contrôle

```
Ecrire "Quel déplacement ?"
Ecrire " - 0: en haut à gauche"
Ecrire " - 1: en haut à droite"
Ecrire " - 2: en bas à gauche"
Ecrire " - 3: en bas à droite"
Correct ← Faux
TantQue Non Correct
  Lire Dep
  Si Dep >= 0 et Dep <= 3 Alors
    Correct ← Vrai
  FinSi
FinTantQue
```

i2 et j2 sont les futures coordonnées du pion. La variable booléenne MoveOK vérifie la validité de ce futur emplacement

```
i2 ← posi + Mouv(Dep, 0)
j2 ← posj + Mouv(Dep, 1)
MoveOK ← i2 >= 0 et i2 <= 7 et j2 >= 0 et j2 <= 7
```

Cas où le déplacement est valide

```
Si MoveOK Alors
  Damier(posi, posj) ← Faux
  Damier(i2, j2) ← Vrai
```

Affichage du nouveau damier

```
  Pour i ← 0 à 7
    Pour j ← 0 à 7
      Si Damier(i, j) Alors
```

```
        Ecrire " O ";
      Sinon
        Ecrire " X ";
      FinSi
    j suivant
    Ecrire ""
  i suivant
Sinon
```

Cas où le déplacement n’est pas valide

```
  Ecrire "Mouvement impossible"
FinSi
Fin
```

PARTIE 9
Enoncé des Exercices

Exercice 9.1

Parmi ces affectations (considérées indépendamment les unes des autres), lesquelles provoqueront des erreurs, et pourquoi ?

```
Variables A, B, C en Numérique
Variables D, E en Caractère
A ← Sin(B)
A ← Sin(A + B * C)
B ← Sin(A) – Sin(D)
```

```
D ← Sin(A / B)
C ← Cos(Sin(A)
```

Exercice 9.2

Ecrivez un algorithme qui demande un mot à l'utilisateur et qui affiche à l'écran le nombre de lettres de ce mot (c'est vraiment tout bête).

Exercice 9.3

Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui affiche à l'écran le nombre de mots de cette phrase. On suppose que les mots ne sont séparés que par des espaces (et c'est déjà un petit peu moins bête).

Exercice 9.4

Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui affiche à l'écran le nombre de voyelles contenues dans cette phrase.

On pourra écrire deux solutions. La première déploie une condition composée bien fastidieuse. La deuxième, en utilisant la fonction Trouve, allège considérablement l'algorithme.

Exercice 9.5

Ecrivez un algorithme qui demande une phrase à l'utilisateur. Celui-ci entrera ensuite le rang d'un caractère à supprimer, et la nouvelle phrase doit être affichée (on doit réellement supprimer le caractère dans la variable qui stocke la phrase, et pas uniquement à l'écran).

Exercice 9.6 - Cryptographie 1

Un des plus anciens systèmes de cryptographie (aisément déchiffrable) consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les A deviennent des B, les B des C, etc. Ecrivez un algorithme qui demande une phrase à l'utilisateur et qui la code selon ce principe. Comme dans le cas précédent, le codage doit s'effectuer au niveau de la variable stockant la phrase, et pas seulement à l'écran.

Exercice 9.7 - Cryptographie 2 - *le chiffre de César*

Une amélioration (relative) du principe précédent consiste à opérer avec un décalage non de 1, mais d'un nombre quelconque de lettres. Ainsi, par exemple, si l'on choisit un décalage de 12, les A deviennent des M, les B des N, etc.

Réalisez un algorithme sur le même principe que le précédent, mais qui demande en plus quel est le décalage à utiliser. Votre sens proverbial de l'élégance vous interdira bien sûr une série de vingt-six "Si...Alors"

Exercice 9.8 - Cryptographie 3

Une technique ultérieure de cryptographie consista à opérer non avec un décalage systématique, mais par une substitution aléatoire. Pour cela, on utilise un alphabet-clé, dans lequel les lettres se succèdent de manière désordonnée, par exemple :

HYLUJPVREAKBNDOFSQZCWMGITX

C'est cette clé qui va servir ensuite à coder le message. Selon notre exemple, les A deviendront des H, les B des Y, les C des L, etc.

Ecrire un algorithme qui effectue ce cryptage (l'alphabet-clé sera saisi par l'utilisateur, et on suppose qu'il effectue une saisie correcte).

Exercice 9.9 - Cryptographie 4 - *le chiffre de Vigenère*

Un système de cryptographie beaucoup plus difficile à briser que les précédents fut inventé au XVIe siècle par le français Vigenère. Il consistait en une combinaison de différents chiffres de César.

On peut en effet écrire 25 alphabets décalés par rapport à l'alphabet normal :

- l'alphabet qui commence par B et finit par …YZA
- l'alphabet qui commence par C et finit par …ZAB
- etc.

Le codage va s'effectuer sur le principe du chiffre de César : on remplace la lettre d'origine par la lettre occupant la même place dans l'alphabet décalé.
Mais à la différence du chiffre de César, un même message va utiliser non un, mais plusieurs alphabets décalés. Pour savoir quels alphabets doivent être utilisés, et dans quel ordre, on utilise une clé.
Si cette clé est "VIGENERE" et le message "Il faut coder cette phrase", on procèdera comme suit :
La première lettre du message, I, est la 9e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la première lettre de la clé, V. Dans cet alphabet, la 9e lettre est le D. I devient donc D.
La deuxième lettre du message, L, est la 12e lettre de l'alphabet normal. Elle doit être codée en utilisant l'alphabet commençant par la deuxième lettre de la clé, I. Dans cet alphabet, la 12e lettre est le S. L devient donc S, etc.
Quand on arrive à la dernière lettre de la clé, on recommence à la première.
Ecrire l'algorithme qui effectue un cryptage de Vigenère, en demandant bien sûr au départ la clé à l'utilisateur.

Exercice 9.10

Ecrivez un algorithme qui demande un nombre entier à l'utilisateur. L'ordinateur affiche ensuite le message "Ce nombre est pair" ou "Ce nombre est impair" selon le cas.

Exercice 9.11

Ecrivez les algorithmes qui génèrent un nombre Glup aléatoire tel que …

- 0 =< Glup < 2
- –1 =< Glup < 1

- 1,35 =< Glup < 1,65
- Glup émule un dé à six faces
- -10,5 =< Glup < +6,5
- Glup émule la somme du jet simultané de deux dés à six faces

Corrigés des Exercices

Exercice 9.1

```
A ← Sin(B)              Aucun problème
A ← Sin(A + B * C)      Aucun problème
B ← Sin(A) – Sin(D)     Erreur ! D est en caractère
D ← Sin(A / B)          Aucun problème… si B est différent de zéro
C ← Cos(Sin(A)          Erreur ! Il manque une parenthèse fermante
```

Exercice 9.2

Vous étiez prévenus, c'est bête comme chou ! Il suffit de se servir de la fonction Len, et c'est réglé :

```
Variable Mot en Caractère
Variable Nb en Entier
Debut
Ecrire "Entrez un mot : "
Lire Mot
Nb ← Len(Mot)
Ecrire "Ce mot compte ", Nb, " lettres"
Fin
```

Exercice 9.3

Là, on est obligé de compter par une boucle le nombre d'espaces de la phrase, et on en déduit le nombre de mots. La boucle examine les caractères de la phrase un par un, du premier au dernier, et les compare à l'espace.

```
Variable Bla en Caractère
Variables Nb, i en Entier
Debut
Ecrire "Entrez une phrase : "
Lire Bla
Nb ← 0
Pour i ← 1 à Len(Bla)
  Si Mid(Bla, i , 1) = " " Alors
    Nb ← Nb + 1
  FinSi
i suivant
Ecrire "Cette phrase compte ", Nb + 1, " mots"
Fin
```

Exercice 9.4

Solution 1 : pour chaque caractère du mot, on pose une très douloureuse condition composée. Le moins que l'on puisse dire, c'est que ce choix ne se distingue pas par son élégance. Cela dit, il marche, donc après tout, pourquoi pas.

```
Variable Bla en Caractère
Variables Nb, i, j en Entier
Debut
Ecrire "Entrez une phrase : "
Lire Bla
Nb ← 0
Pour i ← 1 à Len(Bla)
```

```
  Si Mid(Bla, i, 1) = "a" ou Mid(Bla, i, 1) = "e" ou Mid(Bla, i, 1) = "i" ou
Mid(Bla, i, 1) = "o" ou Mid(Bla, i, 1) = "u" ou Mid(Bla, i, 1) = "y" Alors
    Nb ← Nb + 1
  FinSi
i suivant
Ecrire "Cette phrase compte ", Nb, " voyelles"
Fin
```

Solution 2 : on stocke toutes les voyelles dans une chaîne. Grâce à la fonction Trouve, on détecte immédiatement si le caractère examiné est une voyelle ou non. C'est nettement plus sympathique…

```
Variables Bla, Voy en Caractère
Variables Nb, i, j en Entier
Debut
Ecrire "Entrez une phrase : "
Lire Bla
Nb ← 0
Voy ← "aeiouy"
Pour i ← 1 à Len(Bla)
  Si Trouve(Voy, Mid(Bla, i, 1)) <> 0 Alors
    Nb ← Nb + 1
  FinSi
i suivant
Ecrire "Cette phrase compte ", Nb, " voyelles"
Fin
```

Exercice 9.5

Il n'existe aucun moyen de supprimer directement un caractère d'une chaîne… autrement qu'en procédant par collage. Il faut donc concaténer ce qui se trouve à gauche du caractère à supprimer, avec ce qui se trouve à sa droite. Attention aux paramètres des fonctions Mid, ils n'ont rien d'évident !

```
Variable Bla en Caractère
Variables Nb, i, j en Entier
Début
Ecrire "Entrez une phrase : "
Lire Bla
Ecrire "Entrez le rang du caractère à supprimer : "
Lire Nb
L ← Len(Bla)
Bla ← Mid(Bla, 1, Nb – 1) & Mid(Bla, Nb + 1, L – Nb)
Ecrire "La nouvelle phrase est : ", Bla
Fin
```

Exercice 9.6

Sur l'ensemble des exercices de cryptographie, il y a deux grandes stratégies possibles :

- soit transformer les caractères en leurs codes ASCII. L'algorithme revient donc ensuite à traiter des nombres. Une fois ces nombres transformés, il faut les reconvertir en caractères.

- soit en rester au niveau des caractères, et procéder directement aux transformations à ce niveau. C'est cette dernière option qui est choisie ici, et pour tous les exercices de cryptographie à venir.

Pour cet exercice, il y a une règle générale : pour chaque lettre, on détecte sa position dans l'alphabet, et on la remplace par la lettre occupant la position suivante. Seul cas particulier, la

vingt-sixième lettre (le Z) doit être codée par la première (le A), et non par la vingt-septième, qui n'existe pas !

```
Variables Bla, Cod, Alpha en Caractère
Variables i, Pos en Entier
Début
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
Pour i ← 1 à Len(Bla)
  Let ← Mid(Bla, i, 1)
  Si Let <> "Z" Alors
    Pos ← Trouve(Alpha, Let)
    Cod ← Cod & Mid(Alpha, Pos + 1, 1)
  Sinon
    Cod ← Cod & "A"
  FinSi
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.7

Cet algorithme est une généralisation du précédent. Mais là, comme on ne connaît pas d'avance le décalage à appliquer, on ne sait pas a priori combien de "cas particuliers", à savoir de dépassements au-delà du Z, il va y avoir.

Il faut donc trouver un moyen simple de dire que si on obtient 27, il faut en réalité prendre la lettre numéro 1 de l'alphabet, que si on obtient 28, il faut en réalité prendre la numéro 2, etc. Ce moyen simple existe : il faut considérer le reste de la division par 26, autrement dit le modulo.

Il y a une petite ruse supplémentaire à appliquer, puisque 26 doit rester 26 et ne pas devenir 0.

```
Variable Bla, Cod, Alpha en Caractère
Variables i, Pos, Décal en Entier
Début
Ecrire "Entrez le décalage à appliquer : "
Lire Décal
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
Pour i ← 1 à Len(Bla)
  Let ← Mid(Bla, i, 1)
  Pos ← Trouve(Alpha, Let)
  NouvPos ← Mod(Pos + Décal, 26)
  Si NouvPos = 0 Alors
    NouvPos ← 26
  FinSi
  Cod ← Cod & Mid(Alpha, NouvPos, 1)
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.8

Là, c'est assez direct.

```
Variable Bla, Cod, Alpha en Caractère
Variables i, Pos, Décal en Entier
Début
Ecrire "Entrez l'alphabet clé : "
Lire Clé
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
Pour i ← 1 à Len(Bla)
  Let ← Mid(Bla, i, 1)
  Pos ← Trouve(Alpha, Let)
  Cod ← Cod & Mid(Clé, Pos, 1)
i Suivant
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.9

Le codage de Vigenère n'est pas seulement plus difficile à briser; il est également un peu plus raide à programmer. La difficulté essentielle est de comprendre qu'il faut deux boucles: l'une pour parcourir la phrase à coder, l'autre pour parcourir la clé. Mais quand on y réfléchit bien, ces deux boucles ne doivent surtout pas être imbriquées. Et en réalité, quelle que soit la manière dont on l'écrit, elle n'en forment qu'une seule.

```
Variables Alpha, Bla, Cod, Clé, Let en Caractère
Variables i, Pos, PosClé, Décal en Entier
Début
Ecrire "Entrez la clé : "
Lire Clé
Ecrire "Entrez la phrase à coder : "
Lire Bla
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
Cod ← ""
PosClé ← 0
Pour i ← 1 à Len(Bla)
```

On gère la progression dans la clé. J'ai effectué cela "à la main" par une boucle, mais un joli emploi de la fonction Modulo aurait permis une programmation en une seule ligne!

```
  PosClé ← PosClé + 1
  Si PosClé > Len(Clé) Alors
    PosClé ← 1
  FinSi
```

On détermine quelle est la lettre clé et sa position dans l'alphabet

```
  LetClé ← Mid(Clé, PosClé, 1)
  PosLetClé ← Trouve(Alpha, LetClé)
```

On détermine la position de la lettre à coder et le décalage à appliquer. Là encore, une solution alternative aurait été d'employer Mod : cela nous aurait épargné le Si…

```
  Let ← Mid(Bla, i, 1)
  Pos ← Trouve(Alpha, Let)
  NouvPos ← Pos + PosLetClé
  Si NouvPos > 26 Alors
    NouvPos ← NouvPos – 26
  FinSi
  Cod ← Cod & Mid(Alpha, NouvPos, 1)
i Suivant
```

```
Bla ← Cod
Ecrire "La phrase codée est : ", Bla
Fin
```

Exercice 9.10

On en revient à des choses plus simples…

```
Variable Nb en Entier
Ecrire "Entrez votre nombre : "
Lire Nb
Si Nb/2 = Ent(Nb/2) Alors
  Ecrire "Ce nombre est pair"
Sinon
  Ecrire "Ce nombre est pair"
FinSi
Fin
```

Exercice 9.11

```
a) Glup ← Alea() * 2
b) Glup ← Alea() * 2 - 1
c) Glup ← Alea() * 0,30 + 1,35
d) Glup ← Ent(Alea() * 6) + 1
e) Glup ← Alea() * 17 – 10,5
f) Glup ← Ent(Alea()*6) + Ent(Alea()*6) + 2
```

PARTIE 10
Enoncé des Exercices

Exercice 10.1

Quel résultat cet algorithme produit-il ?

```
Variable Truc en Caractère
Début
Ouvrir "Exemple.txt" sur 5 en Lecture
Tantque Non EOF(5)
  LireFichier 5, Truc
  Ecrire Truc
```

```
FinTantQue
Fermer 5
Fin
```

Exercice 10.2

Ecrivez l'algorithme qui produit un résultat similaire au précédent, mais le fichier texte "Exemple.txt" est cette fois de type délimité (caractère de délimitation : /). On produira à l'écran un affichage où pour des raisons esthétiques, ce caractère sera remplacé avec des espaces.

Exercice 10.3

On travaille avec le fichier du carnet d'adresses en champs de largeur fixe.

Ecrivez un algorithme qui permet à l'utilisateur de saisir au clavier un nouvel individu qui sera ajouté à ce carnet d'adresses.

-

Exercice 10.4

Même question, mais cette fois le carnet est supposé être trié par ordre alphabétique. L'individu doit donc être inséré au bon endroit dans le fichier.

Exercice 10.5

Ecrivez un algorithme qui permette de modifier un renseignement (pour simplifier, disons uniquement le nom de famille) d'un membre du carnet d'adresses. Il faut donc demander à l'utilisateur quel est le nom à modifier, puis quel est le nouveau nom, et mettre à jour le fichier. Si le nom recherché n'existe pas, le programme devra le signaler.

Exercice 10.6

Ecrivez un algorithme qui trie les individus du carnet d'adresses par ordre alphabétique.

Exercice 10.7

Soient Toto.txt et Tata.txt deux fichiers dont les enregistrements ont la même structure. Ecrire un algorithme qui recopie tout le fichier Toto dans le fichier Tutu, puis à sa suite, tout le fichier Tata (concaténation de fichiers).

Exercice 10.8

Ecrire un algorithme qui supprime dans notre carnet d'adresses tous les individus dont le mail est invalide (pour employer un critère simple, on considèrera que sont invalides les mails ne comportant aucune arobase, ou plus d'une arobase).

Exercice 10.9

Les enregistrements d'un fichier contiennent les deux champs Nom (chaîne de caractères) et Montant (Entier). Chaque enregistrement correspond à une vente conclue par un commercial d'une société.

On veut mémoriser dans un tableau, puis afficher à l'écran, le total de ventes par vendeur. Pour simplifier, on suppose que le fichier de départ est déjà trié alphabétiquement par vendeur.

Corrigés des Exercices

Exercice 10.1

Cet algorithme écrit l'intégralité du fichier "Exemple.txt" à l'écran

---

Exercice 10.2

```
Variable Truc en Caractère
Variable i en Entier
Debut
Ouvrir "Exemple.txt" sur 5 en Lecture
Tantque Non EOF(5)
  LireFichier 5, Truc
  Pour i ← 1 à Len(Truc)
    Si Mid(Truc, i, 1) = "/" Alors
      Ecrire " "
    Sinon
      Ecrire Mid(Truc, i, 1)
    FinSi
  i Suivant
FinTantQue
Fermer 5
```

Exercice 10.3

```
Variables Nom * 20, Prénom * 17, Tel * 10, Mail * 20, Lig en Caractère
Debut
Ecrire "Entrez le nom : "
Lire Nom
Ecrire "Entrez le prénom : "
Lire Prénom
Ecrire "Entrez le téléphone : "
Lire Tel
Ecrire "Entrez le nom : "
Lire Mail
Lig ← Nom & Prénom & Tel & Mail
Ouvrir "Adresse.txt" sur 1 pour Ajout
EcrireFichier 1, Lig
Fermer 1
Fin
```

Exercice 10.4

Là, comme indiqué dans le cours, on passe par un tableau de strutures en mémoire vive, ce qui est la technique la plus fréquemment employée. Le tri - qui est en fait un simple test - sera effectué sur le premier champ (nom).

```
Structure Bottin
  Nom en Caractère * 20
```

```
  Prénom en Caractère * 15
  Tel en Caractère * 10
  Mail en Caractère * 20
Fin Structure
Tableau Mespotes() en Bottin
Variables MonPote, Nouveau en Bottin
Variables i, j en Numérique
Debut
Ecrire "Entrez le nom : "
Lire Nouveau.Nom
Ecrire "Entrez le prénom : "
Lire Nouveau.Prénom
Ecrire "Entrez le téléphone : "
Lire Nouveau.Tel
Ecrire "Entrez le mail : "
```

Lire Nouveau.Mail

On recopie l'intégralité de "Adresses" dans MesPotes(). Et après tout, c'est l'occasion : quand on tombe au bon endroit, on insère subrepticement notre nouveau copain dans le tableau.

```
Ouvrir "Adresse.txt" sur 1 pour Lecture
i ← -1
inséré ← Faux
Tantque Non EOF(1)
  i ← i + 1
  Redim MesPotes(i)
  LireFichier 1, MonPote
  Si MonPote.Nom > Nouveau.Nom et Non Inséré Alors
    MesPotes(i) ← Nouveau
    Inséré ← Vrai
    i ← i + 1
    Redim MesPotes(i)
  FinSi
  MesPotes(i) ← MonPote
FinTantQue
Fermer 1
```

Et le tour est quasiment joué. Il ne reste plus qu'à rebalancer tel quel l'intégralité du tableau MesPotes dans le fichier, en écrasant l'ancienne version.

```
Ouvrir "Adresse.txt" sur 1 pour Ecriture
Pour j ← 0 à i
  EcrireFichier 1, MesPotes(j)
j suivant
Fermer 1
Fin
```

Exercice 10.5

C'est un peu du même tonneau que ce qu'on vient de faire, à quelques variantes près. Il y a essentiellement une petite gestion de flag pour faire bonne mesure.

```
Structure Bottin
  Nom en Caractère * 20
  Prénom en Caractère * 15
  Tel en caractère * 10
  Mail en Caractère * 20
Fin Structure
Tableau Mespotes() en Bottin
```

```
Variables MonPote en Bottin
Variables Ancien, Nouveau en Caractère*20
Variables i, j en Numérique
Variable Trouvé en Booléen
Debut
Ecrire "Entrez le nom à modifier : "
Lire Ancien
Ecrire "Entrez le nouveau nom : "
Lire Nouveau
```

On recopie l'intégralité de "Adresses" dans Fic, tout en recherchant le clampin. Si on le trouve, on procède à la modification.

```
Ouvrir “Adresse.txt” sur 1 pour Lecture
i ← -1
Trouvé ← Faux
Tantque Non EOF(1)
  i ← i + 1
  Redim MesPotes(i)
  LireFichier 1, MonPote
  Si MonPote.Nom = Ancien.Nom Alors
    Trouvé ← Vrai
    MonPote.Nom ← Nouveau
  FinSi
  MesPotes(i) ← MonPote
FinTantQue
Fermer 1
```

On recopie ensuite l'intégralité de Fic dans "Adresse"

```
Ouvrir "Adresse.txt" sur 1 pour Ecriture
Pour j ← 0 à i
  EcrireFichier 1, MesPotes(j)
j Suivant
Fermer 1
```

Et un petit message pour finir !

```
Si Trouvé Alors
  Ecrire "Modification effectuée"
Sinon
  Ecrire "Nom inconnu. Aucune modification effectuée"
FinSi
Fin
```

Exercice 10.6

Là, c'est un tri sur un tableau de structures, rien de plus facile. Et on est bien content de disposer des structures, autrement dit de ne se coltiner qu'un seul tableau…

```
Structure Bottin Nom en Caractère * 20
Prénom en Caractère * 15
Tel en caractère * 10
Mail en Caractère * 20
Fin Structure
Tableau Mespotes() en Bottin
Variables Mini en Bottin
Variables i, j en Numérique
Debut
```

On recopie l'intégralité de "Adresses" dans MesPotes…

```
Ouvrir "Adresse.txt" sur 1 pour Lecture
i ← -1
Tantque Non EOF(1)
  i ← i + 1
  Redim MesPotes(i)
  LireFichier 1, MesPotes(i)
FinTantQue
Fermer 1
```

On trie le tableau selon l'algorithme de tri par insertion déjà étudié, en utilisant le champ Nom de la structure :

```
Pour j ← 0 à i - 1
  Mini ← MesPotes(j)
  posmini ← j
  Pour k ← j + 1 à i
    Si MesPotes(k).Nom < Mini.Nom Alors
      mini ← MesPotes(k)
      posmini ← k
    Finsi
  k suivant
  MesPotes(posmini) ← MesPotes(j)
  MesPotes(j) ← Mini
j suivant
```

On recopie ensuite l'intégralité du tableau dans "Adresse"

```
Ouvrir "Adresse.txt" sur 1 pour Ecriture
Pour j ← 0 à i
  EcrireFichier 1, MesPotes(j)
j suivant
Fermer 1
Fin
```

Exercice 10.7

Bon, celui-là est tellement idiot qu'on n'a même pas besoin de passer par des tableaux en mémoire vive.

```
Variable Lig en Caractère
Début
Ouvrir "Tutu.txt" sur 1 pour Ajout
Ouvrir "Toto.txt" sur 2 pour Lecture
Tantque Non EOF(2)
  LireFichier 2, Lig
  EcrireFichier 1, Lig
FinTantQue
Fermer 2
Ouvrir "Tata.txt" sur 3 pour Lecture
Tantque Non EOF(3)
  LireFichier 2, Lig
  EcrireFichier 1, Lig
FinTantQue
Fermer 3
Fermer 1
Fin
```

Exercice 10.8

On va éliminer les mauvaises entrées dès la recopie : si l'enregistrement ne présente pas un mail valide, on l'ignore, sinon on le copie dans le tableau.

```
Structure Bottin
  Nom en Caractère * 20
  Prénom en Caractère * 15
  Tel en caractère * 10
  Mail en Caractère * 20
Fin Structure
Tableau Mespotes() en Bottin
Variable MonPote en Bottin
Variables i, j en Numérique
Debut
```
On recopie "Adresses" dans MesPotes en testant le mail…
```
Ouvrir "Adresse.txt" sur 1 pour Lecture
i ← -1
Tantque Non EOF(1)
  LireFichier 1, MonPote
  nb ← 0
  Pour i ← 1 à Len(MonPote.Mail)
    Si Mid(MonPote.Mail, i, 1) = "@" Alors
      nb ← nb + 1
    FinSi
  i suivant
  Si nb = 1 Alors
    i ← i + 1
    Redim MesPotes(i)
    MesPotes(i) ← MonPote
  FinSi
FinTantQue
Fermer 1
```
On recopie ensuite l'intégralité de Fic dans "Adresse"
```
Ouvrir "Adresse.txt" sur 1 pour Ecriture
Pour j ← 0 à i
  EcrireFichier 1, MesPotes(j)
j Suivant
Fermer 1
Fin
```

Exercice 10.9

Une fois de plus, le passage par un tableau de structures est une stratégie commode. Attention toutefois, comme il s'agit d'un fichier texte, tout est stocké en caractère. Il faudra donc convertir en numérique les caractères représentant les ventes, pour pouvoir effectuer les calculs demandés. Pour le traitement, il y a deux possibilités. Soit on recopie le fichier à l'identique dans un premier tableau, et on traite ensuite ce tableau pour faire la somme par vendeur. Soit on fait le traitement directement, dès la lecture du fichier. C'est cette option qui est choisie dans ce corrigé.
```
Structure Vendeur
  Nom en Caractère * 20
  Montant en Numérique
Fin Structure
Tableau MesVendeurs() en Vendeur
Variables NomPrec * 20, Lig, Nom en caractère
Variables Somme, Vente en Numérique
```
On balaye le fichier en faisant nos additions.

Dès que le nom a changé (on est passé au vendeur suivant), on range le résultat et on remet tout à zéro

```
Debut
Ouvrir "Ventes.txt" sur 1 pour Lecture
i ← -1
Somme ← 0
NomPréc ← ""
Tantque Non EOF(1)
  LireFichier 1, Lig
  Nom ← Mid(Lig, 1, 20)
  Vente ← CNum(Mid(Lig, 21, 10)
  Si Nom = NomPrec Alors
    Somme ← Somme + Vente
  Sinon
    i ← i + 1
    Redim MesVendeurs(i)
    MesVendeurs(i).Nom ← NomPrec
    MesVendeurs(i).Montant ← Somme
    Somme ← 0
    NomPrec ← Nom
  FinSi
FinTantQue
Et n'oublions pas un petit tour de plus pour le dernier de ces messieurs…
i ← i + 1
Redim MesVendeurs(i)
MesVendeurs(i).Nom ← NomPrec
MesVendeurs(i).Montant ← Somme
Fermer 1
Pour terminer, on affiche le tableau à l'écran
Pour j ← 0 à i
  Ecrire MesVendeurs(j)
j suivant
Fin
```


PARTIE 11
Enoncé des Exercices

Exercice 11.1

Écrivez une fonction qui renvoie la somme de cinq nombres fournis en argument.

Exercice 11.2

Écrivez une fonction qui renvoie le nombre de voyelles contenues dans une chaîne de caractères passée en argument. Au passage, notez qu'une fonction a tout à fait le droit d'appeler une autre fonction.

Exercice 11.3

Réécrivez la fonction Trouve, vue précédemment, à l'aide des fonctions Mid et Len (comme quoi, Trouve, à la différence de Mid et Len, n'est pas une fonction indispensable dans un langage).

Corrigés des Exercices

Exercice 11.1

Voilà un début en douceur…

```
Fonction Sum(a, b, c, d, e)
  Renvoyer a + b + c + d + e
FinFonction
```

Exercice 11.2

```
Fonction NbVoyelles(Mot en Caractère)
Variables i, nb en Numérique
Pour i ← 1 à Len(Mot)
  Si Trouve("aeiouy", Mid(Mot, i, 1)) <> 0 Alors
    nb ← nb + 1
  FinSi
i suivant
Renvoyer nb
FinFonction
```

Exercice 11.3

```
Fonction Trouve(a, b)
Variable i en Numérique
Début
i ← 1
TantQue i < Len(a) - Len(b) et b <> Mid(a, i, Len(b))
  i ← i + 1
FinTantQue
Si b <> Mid(a, i, Len(b)) Alors
  Renvoyer 0
Sinon
  Renvoyer i
FinFonction
```

Fonction ChoixDuMot

Quelques explications : on lit intégralement le fichier contenant la liste des mots. Au fur et à mesure, on range ces mots dans le tableau Liste, qui est redimensionné à chaque tour de boucle. Un tirage aléatoire intervient alors, qui permet de renvoyer un des mots au hasard.

```
Fonction ChoixDuMot()
Tableau Liste() en Caractère
Variables Nbmots, Choisi en Numérique
Ouvrir "Dico.txt" sur 1 en Lecture
Nbmots ← -1
Tantque Non EOF(1)
  Nbmots ← Nbmots + 1
```

```
  Redim Liste(Nbmots)
  LireFichier 1, Liste(Nbmots)
FinTantQue
Fermer 1
Choisi ← Ent(Alea() * Nbmots)
Renvoyer Liste(Choisi)
FinFonction
```

Fonction PartieFinie

On commence par vérifier le nombre de mauvaises réponses, motif de défaite. Ensuite, on regarde si la partie est gagnée, traitement qui s'apparente à une gestion de Flag : il suffit que l'une des lettres du mot à deviner n'ait pas été trouvée pour que la partie ne soit pas gagnée. La fonction aura besoin, comme arguments, du tableau Verif, de son nombre d'éléments et du nombre actuel de mauvaises réponses.

```
Fonction PartieFinie(t() en Booleen, n, x en Numérique)
Variables i, issue en Numerique
Si x = 10 Alors
  Renvoyer 2
Sinon
  Issue ← 1
  Pour i ← 0 à n
    Si Non t(i) Alors
      Issue ← 0
    FinSi
  i suivant
  Renvoyer Issue
FinSi
FinFonction
```

Procédure AffichageMot

Une même boucle nous permet de considérer une par une les lettres du mot à trouver (variable m), et de savoir si ces lettres ont été identifiées ou non.

```
Procédure AffichageMot(m en Caractère par Valeur, t() en Booléen par Valeur)
Variable Aff en Caractere
Variable i en Numerique
Aff ← ""
Pour i ← 0 à len(m) - 1
  Si Non t(i) Alors
    Aff ← Aff & "-"
  Sinon
    Aff ← Aff & Mid(mot, i + 1, 1)
  FinSi
i suivant
Ecrire Aff
FinProcédure
```

Remarque : cette procédure aurait également pu être écrite sous la forme d'une fonction, qui aurait renvoyé vers la procédure principale la chaîne de caractères Aff. L'écriture à l'écran de cette chaîne Aff aurait alors été faite par la procédure principale.

Voilà donc une situation où on peut assez indifféremment opter pour une sous-procédure ou pour une fonction.

Procédure SaisieLettre

On vérifie que le signe entré (paramètre b) est bien une seule lettre, qui ne figure pas dans les propositions précédemment effectuées (paramètre a)

```
Procédure SaisieLettre(a, b en Caractère par Référence)
Variable Correct en Booleen
Variable Alpha en Caractere
Début
Correct ← Faux
Alpha ← "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
TantQue Non Correct
  Ecrire "Entrez la lettre proposée : "
  Lire b
  Si Trouve(alpha, b) = 0 ou len(b) <> 1 Alors
    Ecrire "Ce n'est pas une lettre !"
  SinonSi Trouve(a, b) <> 0 Alors
    Ecrire "Lettre déjà proposée !"
  Sinon
    Correct ← Vrai
    a ← a & b
  FinSi
FinTantQue
Fin Procédure
```

Procédure VerifLettre

Les paramètres se multiplient… L est la lettre proposée, t() le tableau de booléens, M le mot à trouver et N le nombre de mauvaises propositions. Il n'y a pas de difficulté majeure dans cette procédure : on examine les lettres de M une à une, et on en tire les conséquences. Le flag sert à savoir si la lettre proposée faisait ou non partie du mot à deviner.

```
Procédure VerifLettre(L, M en Caractère par Valeur, t() en Booléen par Référence, N
en Numérique par Référence)
Variable Correct en Booleen
Début
Correct ← Faux
Pour i ← 1 à Len(M)
  Si Mid(M, i, 1) = L Alors
    Correct ← Vrai
    T(i - 1) ← Vrai
  FinSi
FinTantQue
Si Non Correct Alors
  N ← N + 1
FinSi
Fin Procédure
```

Procédure Epilogue

```
Procédure Epilogue(M en Caractère par Valeur, N en Numérique par Valeur)
Début
Si N = 2 Alors
  Ecrire "Une mauvaise proposition de trop… Partie terminée !"
  Ecrire "Le mot à deviner était : ", M
Sinon
  Ecrire "Bravo ! Vous avez trouvé !"
FinSi
Fin Procédure
```

Procédure Principale

```
Procédure Principale
Variables Lettre, Mot, Propos en Caractere
```

```
Variables g i, MovRep en Numérique
Tableau Verif() en Booleen
Début
Mot ← ChoixDuMot()
Propos ← ""
Lettre ← ""
Redim Verif(Len(Mot)-1)
Pour i ← 0 à Len(Mot)-1
  Verif(i) ← Faux
i suivant
k ← 0
Tantque k = 0
  AffichageMot(Mot, Verif())
  SaisieLettre(Propos, Lettre)
  VerifLettre(Lettre, Mot, Verif(), MovRep)
  k ← PartieFinie(Verif(), len(mot), MovRep)
FinTantQue
Epilogue(Mot, k)
Fin
```